Amendments to the Claims:

1. (Currently Amended) A method of bypassing a null pointer condition check when compiling a source programs comprising ~~comprised of~~:

creating an entry in a fault to target translation table for null pointer conditions that correspond to the null pointer condition check if the null pointer condition check infrequently encounters null pointer conditions; and

eliminating the null pointer condition check, if the null pointer condition check infrequently encounters null pointer conditions.

2. (Previously Presented) The method of claim 1 further comprising:

gathering statistics as to the number of null pointer condition occurrences the null pointer condition check encounters; and

determining an acceptable rate of occurrence.

3. (Currently Amended) The method of claim 1 further comprising:

responsive to a fault [[a]] that corresponds to a null pointer condition, using a handler program to direct the fault to a target indicated by fault to target translation data.

4. (Currently Amended) The method of claim 2 further comprising:

passing the fault to target translation data from the fault to target translation table to a compiler using a handler program.

5. (Currently Amended) The method of claim 1 further comprising:

compiling the source program to an executable program; and

accessing the fault to target translation table during the compiling of the source program.

6–32. (Cancelled)

33. (Currently Amended) A method for bypassing null pointer condition checks, the method comprising:

selectively eliminating from code null pointer condition checks that encounter null pointer conditions less frequently than a given threshold according to profile feedback for the code; and

installing entries for the selectively eliminated null pointer condition check in a table, wherein the installed entries are utilized to direct execution of the code to code that handles ~~handle~~ null pointer conditions.

34. (Previously Presented) The method of claim 33 further comprising identifying the null pointer condition checks that encounter null pointer conditions less frequently than the given threshold.

35. (Previously Presented) The method of claim 33 further comprising extracting information about null pointer condition frequencies from the profile feedback for the code.

36. (Previously Presented) The method of claim 35 further comprising determining those of the null pointer condition checks that have profile feedback information that indicates the null pointer condition checks occur less frequently than the given threshold.

37. (Previously Presented) The method of claim 33 further comprising generating executable code from the code with eliminated null pointer condition checks.

38. (Previously Presented) The method of claim 33 further comprising generating the table to associate faults with respective null pointer condition handling code units.

39. (Currently Amended) The method of claim 37 further comprising populating the table with instruction identifiers of instructions associated with the eliminated null pointer condition checks and the respective ~~ones of the~~ null pointer condition handling code units.

40. (Previously Presented) The method of claim 39 further comprising the table indicating the instruction identifiers for instructions that cause faults corresponding to the eliminated null pointer condition checks and identifiers for the null pointer condition handling code units.

41. (Currently Amended) An apparatus comprising:

memory; and

means for modifying a program to eliminate [[those]] null pointer condition checks that encounter null pointer conditions infrequently according to profile feedback for the program and to direct faults that correspond to the elimination of [[to]] areas of the program that handle null pointer conditions.

42. (Currently Amended) The apparatus of claim 41 further comprising means for maintaining a table that associates faults with the respective ~~ones of the~~ null pointer condition handling areas of the program.

43. (Previously Presented) The apparatus of claim 41 further comprising means for extracting information from the profile feedback that indicates frequency of null pointer condition occurrences.

44. (Currently Amended) A computer program product encoded on one or more machine-readable media, the computer program product comprising:

a first sequence of instructions executable to identify [[those]] null pointer condition checks of a program that encounter null conditions less frequently than a threshold according to profile feedback for the program and to associate code of the program that depends on the identified null pointer condition checks with corresponding program code that handles null pointer conditions; and

a second sequence of instructions executable to eliminate the identified null pointer condition checks.

45. (Currently Amended) The computer program product of claim 44 further comprising the first sequence of instructions executable to populate a table which [[with]] associates [[of]] the code of the program that depends on the identified null pointer condition checks with the corresponding program code that handles null pointer conditions.

46. (Previously Presented) The computer program product of claim 45, wherein the associating comprises associating addresses of the null pointer condition check dependent program code with respective addresses of the null pointer condition handling program code.

47. (Previously Presented) The computer program product of claim 44 further comprising a third sequence of instructions to generate an executable representation of the program with eliminated null condition checks.

48. (Previously Presented) The computer program product of claim 44 further comprising a third sequence of instructions to extract information from the profile feedback that indicates frequency of null pointer conditions.

49. (Cancelled)